KALIDO

Data Warehouse Lifecycle Management

Concepts and Principles

Cliff Longman, CTO, Kalido
April 2004

Table of Contents

Summary

Data warehouses provide accurate, timely management information for large enterprises. In a business world characterized by frequent, far-reaching change, data warehouses quickly lose their relevance if they do not adapt to this change.

When built using traditional software development methodologies, data warehouses are costly, and do not adapt efficiently to change.

Data Warehouse Lifecycle Management (DWLM) is a new discipline for the creation and ongoing management of a data warehouse throughout its operational life. DWLM delivers enterprise-scale data warehouses that adapt efficiently to change, at lower cost than traditional software development methodologies.

DWLM employs two development methods: Rapid Iteration and Federation. These methods enable organizations to implement an enterprise-wide data warehouse in bite-sized pieces.

Successful DWLM can only be achieved with a data warehouse that can react quickly to change: an adaptive data warehouse. Only a data warehouse capable of making structural changes in hours or days rather than the usual weeks or months will support the rapid iteration and federated development methods. While traditionally-constructed data warehouses require IT expertise for any structural changes, adaptive data warehouses give users access to a business model that can be changed without IT expertise, delivering unprecedented flexibility.

"Shell Downstream in Europe uses an adaptive data warehouse to determine profitability of key B2B customers, improving overall profitability and customer service. As a result, the organization has been able to significantly improve its gross margins."

Offering high-speed implementation and faster cycles of change, Data Warehouse Lifecycle Management reduces risk and increases return on investment.

Why Global 2000 Enterprises need Data Warehouses

Every enterprise needs a view of its performance across all its operations. Yet vital performance data is usually spread across multiple operational systems (CRM, ERP, Supply Chain), running on different IT platforms, in different parts of the business, with different physical data structures and different identification schemes. The complexity and cost of capturing and summarizing this data into a meaningful whole often prevents enterprises from getting the clear view of global performance they need.

To overcome these issues, many enterprises use data warehouses to store a copy of data drawn from operational systems using Extract, Transform, and Load (ETL) technology. (See Figure 1 below.) Such data warehouses present data on demand to business users and analysts via business intelligence, reporting and analytical tools.

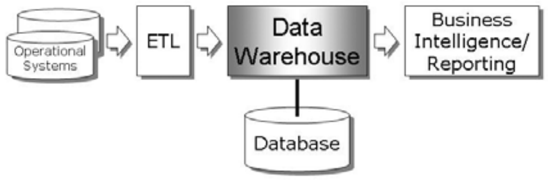

Figure 1: Basic data warehouse components

By acting as a central source for management information, a data warehouse delivers a single version of the truth that spans multiple operations. This gives executives visibility of business performance, improving the organization's ability to react to competitive pressures.

With a data warehouse that delivers consistent management information, an enterprise can react more rapidly than its competitors to new opportunities or downturns in the market.

Gaining a single version of the truth also improves compliance with increasingly rigorous regulatory reporting requirements, such as Sarbanes-Oxley, and can help businesses meet operational standards, such as those demanded by Basel II for financial organizations. Well-constructed data warehouses also help to reduce the impact of future legislative reporting requirements.

Data Warehouse and Business Out Of Step

Unless data warehousing products offered by ERP vendors meet companies' requirements very closely, there has been no other option than to build, rather than buy, data warehouses. In practice, most organizations customize their ERP implementations and data warehouses to fit their businesses. So, whether a data warehouse is built from scratch, or based on a vendor's pre-built start point, the normal practice is to freeze the business model at a point in time, then build a data warehouse to reflect it. Changes taking place during the development of the data warehouse are ignored. Subsequent business changes are restricted, to avoid the need for costly re-design exercises.

During the design of the data warehouse structure, conventional methodology requires the expected output to be clearly defined before any development work is started. Such definitions must be determined with great care, because once the foundations of the data warehouse are laid, changes in design will be costly. User requirements for information, queries and reporting are defined in advance, then 'hardwired' into the fabric of the data warehouse.

With a data warehouse built in this way, any major change – whether a new business requirement, correction of an error in reporting structures, or simply a user request for information in a different form – will require IT experts to hunt through large systems and identify what needs to be altered, and then make complex changes in the relevant places. The cost will be high, both in terms of IT resource and in terms of the time needed for the data warehouse to catch up with the new business requirement.

The real difficulty with data warehouses is that users cannot reasonably be expected to know what they want until they have used the system, particularly where business intelligence is concerned. In addition, change is a fact of life for any Global 2000 company, whether it is responding to new market conditions or proactively taking a new strategic direction. Performance measurement rapidly loses its relevance if changes in the business are not reflected in the data warehouse and its reporting structures. Even if the enterprise succeeds in defining data warehouse requirements and implementing them, users will come up with new or changed requirements when they start using the warehouse.

Accurate, relevant and timely information can only come from data warehouses, which continually adapt to remain aligned with changing business needs. Data warehouses therefore have their own lifecycles: business users request changes, a data warehouse development team evaluates these changes, translates them into development plans, and finally implements them before turning to the next round of user requests.

The traditional methodology of incorporating changes into new (large-scale) releases of data warehouses makes it difficult to keep data warehouses aligned with the business.

As Frank Buytendijk, Vice President, Gartner, Inc., points out – "Most data warehouse project plans focus entirely on technical aspects of initial implementation and change is thereafter seen as a production issue. This does not reflect reality. Enterprises need to adopt a methodology where both business and IT align themselves during the entire life cycle of the data warehouse."

The need to rebuild data structures, and rewrite the logic that loads and analyzes data, slows down each update cycle of a data warehouse built according to traditional methodology. This non-value-add activity causes data warehouses to lag behind change requests from the business. The IT function becomes increasingly mis-aligned with the business, and users become more and more skeptical about the value of data warehousing. As the gulf between the business and the data warehouse grows, enterprise reporting loses relevance. Often, companies throw more IT resources at the problem in an attempt to keep up with change, so the mis-alignment can be costly as well as obscuring business performance.

As data structures (by which we mean the physical tables in the database) fall further behind the business structures they were designed to mirror, business opportunities may be missed and problems discovered too late. The data warehouse itself becomes more costly and slow to adapt to change, significantly increasing its total cost of ownership and reducing its effective return on investment.

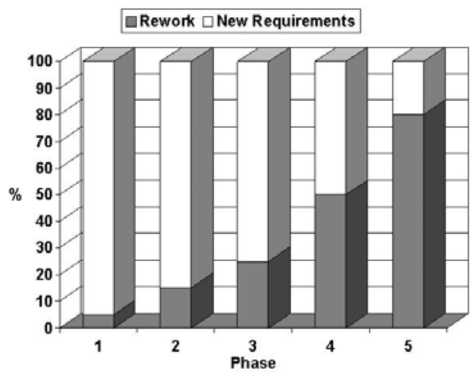

Figure 2: Resource usage for traditionally designed data warehouses.

As the above figure shows, the first phase of rework to a traditionally-designed data warehouse will dedicate almost all the resource to satisfying new requirements. (At the end of each phase, there will usually be some rework to development carried out at the beginning of that phase). Each additional phase, however, will introduce new requirements that demand changes to work completed in previous phases. With a fixed budget, this means that less work is possible in each phase. The alternative is an increasing budget that accommodates the rework as well as the new work.

Changes to the physical database schema are resource-hungry, often demanding data unloading, reformatting, reloading, changes to load and extract code, reworking of data integrity checking and rules, and many other housekeeping tasks such as auditing, backup and recovery. None of this rework adds any value, and its only function is to enable new requirements to sit alongside old ones.

According to the Cutter Consortium: "41 per cent have experienced data warehouse project failures."

By the time four or five versions of a warehouse have been implemented, rework can outweigh work on new requirements, so that the majority of budget is spent on non-value-add activity. In extreme cases, rework takes over entirely, and it becomes impractical to adapt the warehouse for any new requirements. It must then be scrapped or replaced (accounting for the 41% failure rate quoted by Cutter Consortium).

Inflexible by Design

The root cause of this rework is that data is 'hardwired' for a specific use. At the initial design stage, a data warehouse designer will decide the structure of the data warehouse, and then go through a complex implementation to load and analyze data in this 'hardwired' structure. When the business function wants or needs to use the data warehouse differently, the 'hardwired' structure and complex code need to change accordingly. 'Hardwired' data is what makes traditionally designed data warehouses inflexible.

What is needed is an approach that accepts change as the norm, rather than trying to eliminate it or ignore it.

Introducing Data Warehouse Lifecycle Management (DWLM)

Data Warehouse Lifecycle Management is the management of one or more data warehouses throughout their operational life, from initial inception through creation, operation and a lifetime of modification.

DWLM employs an iterative approach to data warehouse development, decreasing the data warehouse cycle time, and can manage a staged roll-out of loosely-coupled data warehouses acting as a single, federated whole. DWLM enables enterprises to align information systems with the changing business operations they are designed to support.

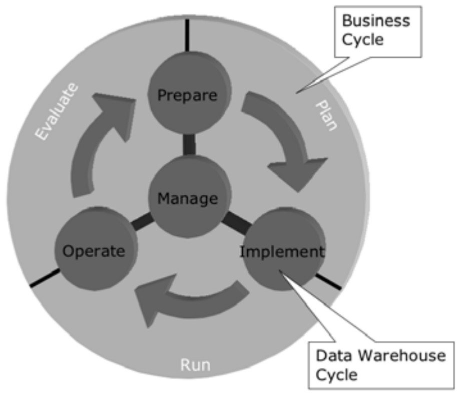

Figure 3: Alignment of business and data warehouse lifecycles.

As Figure 3 on the previous page shows, Data Warehouse Lifecycle Management introduces three phases: 'Prepare', 'Implement', and 'Operate'. These overlap with three phases in the business cycle: 'Plan', 'Run' and 'Evaluate'.

Businesses 'Plan' (along strategic, tactical and operational axes), 'Run' (in terms of operations and management decision-making), and 'Evaluate' their performance in cycles. Sometimes these cycles are tactical and quickly completed (for example a sales promotion on an over-stocked product), sometimes they are strategic (the acquisition of another company). To be of value, a data warehouse must keep in step with these business cycles.

As a business evaluates one cycle and starts planning for the next, the data warehouse lifecycle management 'Prepare' phase configures a data warehouse to capture and report on the relevant data for that business cycle. As the business moves from the 'Plan' to the 'Run' phase, the data warehouse progresses to the 'Implement' phase, in which it is adapted to the requirements of the new cycle and is prepared for operation. As the business moves from 'Run' to 'Evaluate', the 'Operate' phase of the data warehouse collects a record of what the business is doing and analyses it, to monitor business performance. This gives the business information on which to base decisions about the next cycle of change.

In such a dynamic environment, an ongoing lifecycle of change needs to be incorporated into the data warehouse methodology. To do this, Data Warehouse Lifecycle Management introduces two key approaches: Rapid Iteration, and Federation.

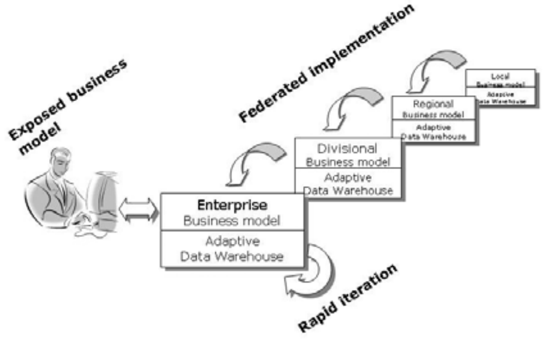

Figure 4: Rapid Iteration and Federation are two key methods contributing to the development of Adaptive Data Warehouses.

Rapid Iteration

Traditionally, IT development groups release versions of a data warehouse periodically, having packaged multiple change requests into discrete development projects. Each project represents disruption to the business, and occupies valuable IT resources, so it is better to delay minor changes and deal with them together with one or two major ones, making a new project worthwhile. This, of course, means that changes to the data warehouse are more risky because of the larger, more complex projects involved. It also means that the data warehouse will lag behind the business, because change requests are not immediately acted upon, but must first accrue into a critical mass.

Following the Data Warehouse Lifecycle Management approach, data warehouses are built for change and so can undergo a greater number of smaller iterations, enabling them to respond quickly to new business requirements. These rapid iterations are not only less risky and resource-intensive, but also enable data warehouses to remain more closely aligned with the business. When frequent small-scale iterations are implemented, data warehouses respond more rapidly to the needs of the business; put simply, the iterative approach expects users to change their minds.

Where a large-scale change to the business does necessitate a new data warehouse release, a flexible data warehouse will make the process faster, less costly and easier than it would be for a traditionally developed data warehouse.

Federation

Complex enterprises benefit from consistent high-level management information and a clear view of how the business is performing. One way to improve the visibility into business operations is to standardize reporting structures and data models from the top down. However, while this delivers consistency, it also impedes the flexibility of local operating units to adapt to changes in their local business environment.

Total standardization is impractical for most enterprises. It costs too much, takes too long and is prone to failure because very few areas within a given business will be simple or uniform enough not to require local customization. While it may be possible to standardize currency codes across an enterprise, it is unlikely that standardized product hierarchies would be either feasible or desirable.

A successful data warehouse must be able to report on local business performance as well as global business performance. The performance of a local confection product sold only in Japan would not be of interest at the global level, but the performance of the confection category as a whole probably would be. Local product performance reporting has to sit alongside accurate global category reporting.

The 'federated' model for data warehousing is one in which a hierarchy of data warehouses can exchange data, business models and reporting structures, to allow local autonomy and customization, but also deliver global control and a degree of standardization. A 'federated data warehouse' consists of a set of data warehouse instances that operate semi-autonomously, are generally geographically or organizationally disparate, but which can be thought of as one large data warehouse. Since a federated data warehouse can be built one step at a time, it offers a 'start small, think big' approach to data warehousing. The federated approach significantly reduces risk in a global roll-out, because each local warehouse is smaller in scope, delivers quickly on local requirements, and can be operated by local business units.

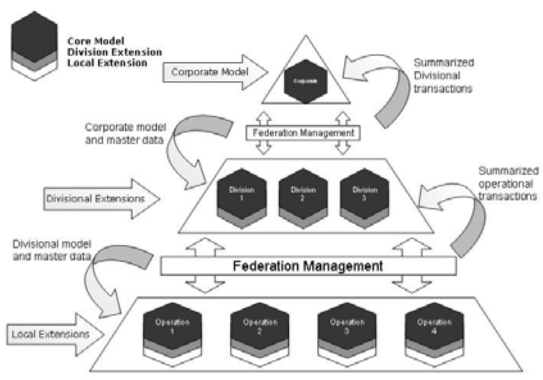

Figure 5: Federation of corporate, divisional, and local warehouses

Early attempts at federated data warehousing used a form of 'virtual' storage, in which the global warehouse did not store a copy of all data; rather, it resided only in the operational systems that created it. Global queries were broken down, run against each system, and the results combined. This approach has been shown to fail, being too complex technically, and placing unpredictable demands for computing resource on operational systems. (Gartner, Inc. labeled this approach 'obsolete before plateau' on their hype curve which tracks new technology from initial inception to maturity on a plateau, when it is adopted as mainstream).

What does work very effectively, however, is a federation of data warehouses, each holding a copy of a core business model and common master data, and where each higher-level data warehouse holds summarized transactions from the level below. Common master data – for example the corporate organization charts – flows downwards from the corporate (global) data warehouse, while summarized transactions – for example, the total number of sports coupes sold in the Fiorentina outlet in Italy - flow upwards from the local data warehouses.

A federation of data warehouses can satisfy both the local need for flexibility and performance, and the global need for consistency and control, with each data warehouse operating independently of all the others.

The federated concept is advantageous not only in terms of daily operations, but also where implementation is concerned. When building a federation of data warehouses that are capable of adapting to change, enterprises can start with a single project, perhaps in an individual country or division of the business, then build up to a global system, adding new data warehouses in order of priority to the business. As long as the data warehouses used are capable of adapting to change, it is not necessary to fix the final architecture of the federation in advance, so a risky monolithic project can be broken down into numerous, low-risk, high-return sub-projects.

In the federated approach, local operating units incur the cost of data warehousing, but achieve payback locally while contributing to the global view, resulting in a sense of ownership and local value. The single, monolithic warehouse approach burdens local operating units with the majority of the cost, but delivers the majority of the benefits at the global level – a project manager's nightmare!

Like iteration, federation is almost impossible to achieve using traditional data warehouse development approaches. If requirements change during implementation (for example, if a division consolidates its ERP infrastructure from five core systems to two) warehouses already in use need to be adapted quickly to the new data sources and structures, so that work can continue.

In a typical global federation project, an organization might implement one data warehouse per group of operating companies plus one per region, and one at the global level. Such a project might take a number of years to complete (it took nearly two years to implement 80 operations in the Shell Downstream business), and because change is bound to be required during the construction of the federation, the federated approach is only practical if the data warehouses used are adaptive data warehouses.

Five Principles for Designing Adaptive Data Warehouses

The Rapid Iteration and Federation — as proposed by data warehouse lifecycle management — demand a data warehouse architecture that is adaptive to change.

An Adaptive Data Warehouse is one that is capable of adapting rapidly to business changes. In the current business environment, this means producing any new analysis from existing information within one working day, and restructuring data to a new business model, or adding new data feeds, within a small number of working days. Major new releases of adaptive data warehouses cause little or no disruption in use and should be implemented in one or two weeks. In terms of speed and effort, this is an order of magnitude better than traditionally-built data warehouses.

With an adaptive data warehouse, the rework curve is quite different from that of the traditional data warehouse, quickly leveling off at an economically acceptable level. An adaptive data warehouse insulates data from the effects of business change, so that data already stored requires no reformatting, and no new or changed loading or extract code is required.

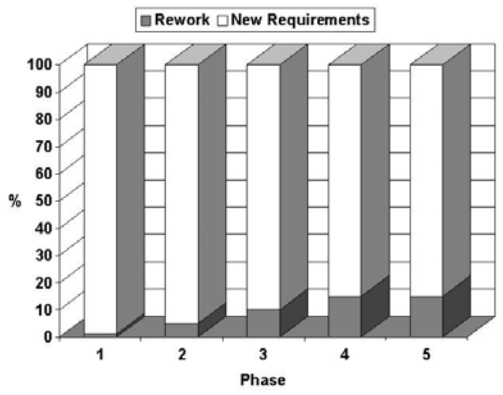

Figure 6: Percentage of resource used to rework earlier phases for adaptive data warehouses

Furthermore, maintenance functions such as purge/archive/restore can also be generic, so they are available automatically as new phases are added, with no new work required. Rework in the adaptive data warehouse is limited to that caused by misunderstandings in requirements and human error, rather than being part of the fabric of the design.

The five flexible design principles outlined below aim to deliver adaptive data warehouses that stay aligned with business needs.

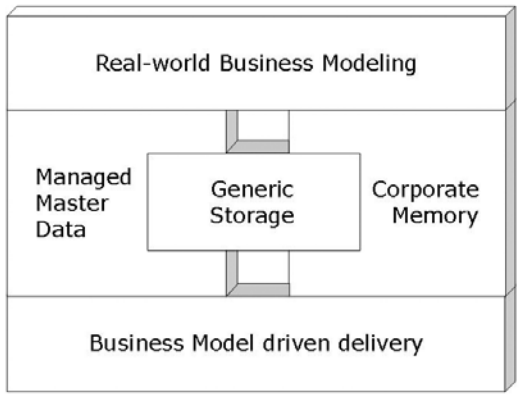

Figure 7: The five DWLM design principles

Principle I: Generic Storage

Data warehouses that set a physical storage structure (database tables and columns) to meet pre-determined business requirements will not be quick to adapt when those requirements change. For an adaptive data warehouse, a generic storage model is crucial.

Much of the cost and time involved in modifying a traditionally designed data warehouse with a pre-set physical storage structure is associated with rewriting SQL, database logic and other code each time the database structure changes. There are three main levels on which change might occur:

- Data - for example, the re-branding of a product
- Data structure - for example, the addition of a new level into an analysis hierarchy, or the addition of a new level at which departmental budgets are apportioned
- Infrastructure – for example, the addition of a new type of data feed, such as market survey data

In an ideal world, it would be possible to adapt to business change without needing to make any changes whatsoever in the physical database structure. There is currently no way of doing this for every type of business change, but a generic storage structure brings the ultimate goal closer.

The most promising design approach to generic storage is the 'reduced schema' approach, which proposes the storage of different types of data in the same table. The same SQL is used to load, access and analyze data, whatever its type, making it much easier to add new types of data and to 'reuse' the system's facilities across data types.

The figure below shows four alternative ways to implement the same requirement (a 3-level organizational hierarchy) physically. In Option 1, there is one table for each type of organization; Option 2 'genericizes' everything below the level of company; Option 3 stores all organizational units in a single table; and Option 4 stores data of all types (not just organizational) in a single table.

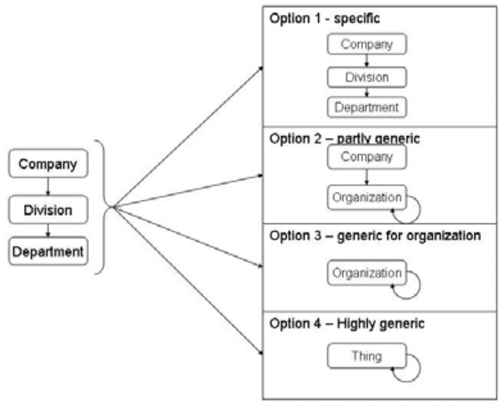

Figure 8: Alternative ways to implement an organizational hierarchy

Each implementation option exhibits different qualities:

Flexibility: Moving from Option 1 to Option 4, the flexibility increases. To extend Option 1 to include a 4th level requires new tables to be created, with all the associated code to load and access them. Option 2 can accommodate a new organizational structure below the level of 'Company' without change. Option 3 can accommodate new organizational structures at any level. Option 4 can accommodate whole new data types beyond just organizational.

Legibility: The more generic options are more difficult to understand than the less generic ones: moving from Option 1 to Option 4, it becomes harder to determine what is in the tables just by looking at them. The software written to load and access the data becomes more obscure as the design becomes more generic.

Volume of software required: More generic structures require less software to make them work. Option 1 requires specific software to load each table, whereas in Option 4, the same software will work for loading all types of data (regardless of how many different types there are). The same is true of software to access and manipulate the data – the more generic the option, the less software is required.

Performance tuning: It is possible to tune Option 1 for very specific operations (e.g. to make an operation on departments particularly efficient). The more generic the implementation option, the less scope there will be for tuning specific operations. Tuning on the more generic implementations does, however, benefit database operations on data of all the types covered by that option.

So, while delivering the desired flexibility, the highly generic approach (Option 4) to storage has its disadvantages, principally:

- Poor legibility of the database structure for humans
- Difficulty of capturing rules and constraints
- Impaired performance

Legibility (to software developers) is a concern, because in a generic data store the meaning of the data is no longer self-evident from table and column names. This problem can be resolved through the inclusion of sufficient metadata in an additional sub-system designed to document the meaning of the generically-stored data. In a metadata sub-system, the meaning of the generically-stored data is itself held as data. As a simple example, a separate column could be added to the THING table in Option 4 to hold the TYPE of each row (indicating whether the row was a COMPANY, DIVISION, DEPARTMENT, or any other type). Storing the metadata separately has the effect of making it far easier for business users (rather than IT developers) to own and maintain names and definitions of the business data, improving control and freeing up IT resources. In very advanced generic storage systems, the metadata is itself stored generically, permitting a further degree of flexibility.

From a technical point of view, it may appear that generic data storage will impair the performance of the data warehouse. Empirical evidence suggests otherwise: commercially-available relational databases have been shown to perform as well, or very nearly as well, with generic designs as with traditional designs. Where performance is an absolute priority, a hybrid design may be considered, in which the performance-sensitive portions of the data warehouse are handled conventionally, while the rest of the data is handled generically. Although this will not provide optimal flexibility, it will certainly improve on current data storage practices.

An alternative hybrid solution supports high-performance access by creating replication facilities that manage a copy of the generically-stored data in a less generic form (e.g. a star schema). For a limited capital outlay – the cost of additional storage and of developing the replication manager – changes in the generic data store can be reflected in the less-generic structure, so that at least some of the advantages of the fully-generic approach are retained.

Generic data storage also allows multiple perspectives of the data to co-exist. Where two different countries (or business functions, or product groups, etc.) have a different view of the data (perhaps reflected as different views of product or customer), both of these views can, and should, be supported in a data warehouse.

Allowing multiple classifications and perspectives to co-exist within a federation of data warehouses enables each country (or division or operation, etc.) to analyze its world in a familiar way, helping to reduce resistance to enterprise-level data warehousing. A top-level global warehouse introduces a standardized corporate view for senior management, in which all products have a single coding, and are integrated with all customers, across all geographies. Without generic storage, traditionally designed data warehouses struggle to reach this balance between global and local needs. The result is either a lack of clarity at the top level, or (more frequently) the enforcement of rigid, unsuitable global standards at local levels.

Principle II: Data Warehouse Delivery-driven by the Business Model

In large companies the scope of coverage required from a data warehouse constantly increases, which (as a result) leads to increased complexity of this data warehouse. In a traditional data warehousing environment, the IT function is in charge of altering the physical structure of the database as it grows larger and more complex. The business function will understand what needs to change at the level of the business model, while the IT function will understand what is needed at the level of the tables and columns in the database. There is often a complex mapping between the two levels, introducing the risk of error, and requiring a costly process of translation between them.

If it were possible for business users to manage change in a data warehouse themselves, this would eliminate the need for lengthy consultation between the business and technical functions, and enable data warehouses to be far more adaptive. Even where only a limited IT knowledge is required, the business function is always looking for more productive and less error-prone ways to keep a data warehouse current. The ideal solution is to raise the level at which a data warehouse is configured, going from the physical database level to the level of the business model. When this is done, business users can modify data warehouses themselves without resorting to highly skilled IT development resources. It should be possible, for example, for a financial controller to change the definition of net profit and to add some new budget data to a data warehouse, confident that the results produced will carry out currency conversion and time variance and all the other complexities correctly.

The acid test for a data warehouse operating at the business level is that it should be possible to build and operate it without anyone ever knowing the names of the tables in which the data is stored. The data warehouse should handle the conversion of the business model to database structure, and provide facilities – abstracted to the business model level – for loading and querying the database.

Ideally, people running the data warehouse should only ever have to operate at the business level.

Principle III: Managed Master Data

Master (or reference) Data is data about products, customers, the organization, geography, and so on. Any data that can be referenced is master data, and the more often a piece of data is referenced, the more important it is to manage it. When it is properly managed, master data provides a consistent context against which to measure business performance. For example, the true global sales of a product category (e.g. 'industrial lubricants') will be reported inaccurately if a product sold in Brazil is classified as 'industrial lubricant', but the same product in Japan is classified 'commercial lubricant' . Poor management of master data makes it much harder to gain a clear view of how a business is performing.

In large organizations, it is insufficient to use transactional systems to manage master data because they were not designed to do so, and only rarely does a single transaction system hold all the master data for a single topic (such as product).

For data warehouses in large-scale businesses – especially complex, global businesses with many products and services, geographically dispersed, and operating under multiple cultures and regulatory systems – it is especially important to take control of master data.

In an ideal situation, all master data would be consistent and accurate across the entire business, including suppliers, customers and other parties the business interacts with. In practice, most organizations struggle to achieve consistency in their master data.

So an important component of an adaptive data warehouse is a scheme for managing master data. Such a scheme needs to recognize that the data is not currently well-organized or easily-accessible. It needs to put in place a process for the ongoing management of master data, giving the business owners of the data the tools to improve, enrich, authorize and publish master data in a form acceptable to the numerous systems used to run the business, including the data warehouses used to manage business performance. Without well-managed master data, data warehouses produce questionable results, which significantly diminish their utility in running a business.

In addition to creating a reliable context for transaction data in a data warehouse, the effective management of master data also improves data quality in all systems. By encouraging data owners to focus on their local coding schemes, master data management can help improve consistency and thereby reduce errors in operations, such as goods being delivered to the wrong addresses.

Principle IV: Real-World Business Modeling

A Business Model is a model of a business requirement from which a data warehouse can be designed. The more accurately such a business model reflects the real world, the less it will need changing and the less a data warehouse built from it will need changing. Consequently the more closely the data warehouse will match the business need.

Taking the time at the planning stage to create an accurate business model pays dividends during implementation and operation. Changing a business model could take only a few moments of discussion, while changing a database and all the corresponding code to maintain and analyze it could take thousands of man hours.

Work on business modeling has been placed in the public domain recently, one example being the ISO 15926 standards in the process plant arena.

At a simple level, we achieve more accurate real-world business models by modeling the underlying nature of each object, rather than its use or role (an adaptive data warehouse can include secondary classification by roles for other purposes). Taking this approach increases the model's durability, because things change their underlying nature infrequently (whereas what they are used for changes very frequently).

Conceptually, 'Customer' and 'Supplier' are subsets of 'Organization' in any business that sells to other companies. If your business model includes 'Customer' and 'Supplier', rather than just 'Organization', then it will be vulnerable to change. For example, a particular supplier might also become a customer, causing the same organization to be classified under two different areas of your business model. With such organizations represented as two independent facets ('Customer' and 'Supplier') in the data warehouse, it will be difficult to gain an accurate picture of the external world or to report accurately. The industry has coined the phrase "single view of customer" to accompany its Customer Relationship Management initiatives, when in fact this would be better stated as "single view of organization", "single view of person" and so on.

When creating an accurate real-world business model, it helps to ensure that each entity can be identified as a member of its type, without reference to any other information. If you need to look at an invoice to know that a particular entity is a 'Customer', this is an indication that 'Customer' is not the underlying concept, but is likely to be a role. You should probably model it as an 'Organization'. As another example, consider Marilyn Monroe. In a real-world model, she would be a 'PERSON'. You may know her as an actress, but that is a role she played for a period of her life. There was a time where she was not (yet) an actress, but she was always a person.

For every entity type in your business model, it is important to ask, "Have I correctly identified and appropriately named the thing itself, or have I merely labeled some role it is currently playing?" For every attribute, you should ask, "Is this really a thing in its own right?"

Frequently, new business requirements arise when a business entity is given a new use or role, for example, when an organization that currently supplies you with goods buys something from you, and so becomes a customer. Equally, if you change from selling solely to companies, and start selling to individual consumers, then some of the entities in your "Customer" table will be companies, some will be individuals, and some may even be your own employees. Attributes that apply to individuals, such as gender, don't apply to companies, and suddenly the structure of the customer table becomes highly complex.

If a data warehouse is designed around the roles played by entities, this will increase the levels of change experienced, and will entail frequent rework to the business model and the data warehouse. If, on the other hand, a data warehouse is designed around the underlying nature of each entity, this decreases the degree of change experienced, since any organization (or person, etc.) remains an organization (or person, etc.) whatever role it is currently playing. Real-world modeling reduces the number of structural data warehouse changes required to cope with changing business models.

Principle V: Time Variance/ Corporate Memory

Organizations generally design data warehouses to be operational over long periods of time. During this time, some real-world objects about which the data warehouse holds data will change numerous times. Unless this history is captured, the data warehouse cannot provide information about the way these objects were in the past, making it difficult or impossible for the business to compare like with like over time.

Time variance is defined as the ability to explicitly record a period of time against an item of data to indicate the period for which that item of data is considered effective.

There are cases where time variance is an important part of the data needed for a business process (for example, where salespeople are paid 5% commission on the amount of 'industrial product' they sell. The commission should be paid on the classification of products at the time of the sale, not the classification at the time at which the commission is calculated). There are also many 'long running' processes during which changes may occur – for example, a container ship may change ownership and insurer (possibly more than once) during a six week voyage.

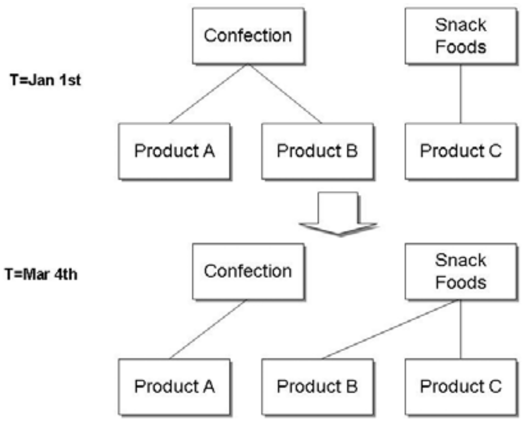

Figure 9: Product reclassification without time variance.

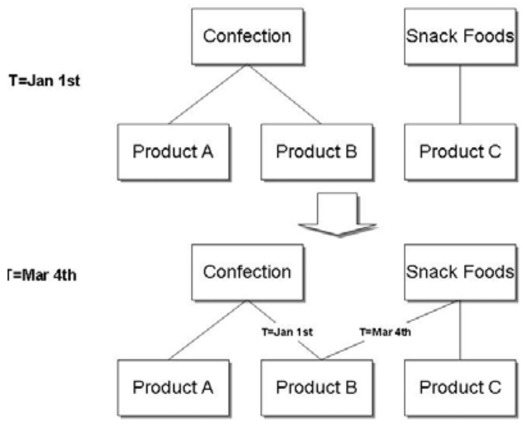

Figure 10: Product reclassification with time variance.

The above figures show an example business scenario – the reclassification of a product – with and without time variance.

Without time variance, Product B only ever has a single category, though on March 4th, it is moved from the Confection category to the Snack foods category. The association between Product B and its former category is lost when the reclassification takes place, so it is impossible to make any meaningful comparisons between categories over time using anything but the current structure. Previously calculated summaries at the category level would have to be re-stated after the category change.

With time variance, however, Product B has two categories: it is listed as confection for T= January 1st, and as snack foods for T= March 4th. It is now possible to generate a true comparison of confection sales at any point in time, taking the re-classification of Product B into account.

So why not apply time variance to everything? In an ideal world, this would be advantageous, but there are practical reasons why doing so is rarely considered. There is a good deal of technical complexity in applying time variance to a relationship or attribute, and the relationships that tend to need time variance are the ones that determine structure, such as how the enterprise was organized, and how products were grouped. There is plenty of scope for technical and logical errors, as well as increasing the resource required to develop and maintain the data warehouse. As a consequence, data warehouse developers usually apply time variance selectively, rather than universally.

It is worth noting that the more generic the storage model is, the easier it becomes to apply time variance universally. In the ultimate case of a single generic database table, time variance can be made to apply to everything uniformly by implementing it once on that single table.

When time variance is applied selectively, the first, and most obvious, selection that can be made is between transaction data, master data and metadata. Transaction data is data about activities that took place at a point in time such as a sale, an invoice, or a payment. Master (or reference) data is common shared data about business constructs such as products, customers, the organization, and so on. Metadata is data that defines or describes other data, such as the definition for Net Profit.

Since transaction data does not change (by definition, it happened at a point in time), nothing is lost in practice by having no time variance on transactions.

Master data changes all the time, and data warehouses analyze transactions against master data (e.g. sales by customer group by region by quarter). To provide accurate and complete coverage of current and potential future requirements, a data warehouse must apply time variance to master data. In fact it is the relationships between items of master data (customer, product, organization, etc.) that are the most valuable place to deal with time variance. These relationships record hierarchies used to roll up transactions for analysis (e.g. organization structure, product categories, geographical breakdown). Time-variant hierarchies are crucial for comparing the performance of a structure in one time period against the same structure in a previous time period, while allowing the structure to change in between. Indeed, time variance for master data is vital if the enterprise needs to summarize any information using associations that may change over the period during which transactions contributed to the summary. Historical records recorded by time-variant data are a significant advantage to many compliance initiatives, since they record the precise state of a business at any given point in time.

If time variance is handled correctly, future dates can be treated in the same way as past dates. This enables time-variant data to be ‘prepared’ and stamped with a future date. Once ready, it can simply be left in the data warehouse, to become effective when that date arrives. When the future date is reached, no data re-organization is needed, so the data warehouse will not require any downtime for the time-variant data to become effective. Transactions recorded in the past can be summarized according to planned future hierarchies, by using master data with future time-stamps.

Finally, there is the question of metadata. Metadata is a record of the meaning of the data, so the time variance of metadata is a record of how this meaning has changed. To have a record of a data warehouse’s definitions as they change over time is a huge advantage – enabling changes in the data warehouse itself to be tracked – but it is rare in practice because data warehouses are normally designed with the (current) metadata enshrined in the database structure.

No commonly-used database management systems offer the ability to view historical metadata alongside the master and transactional data (in their historical context). If the metadata is itself held as (generic) data, then time variance can be applied to it, and a record of the meaning ascribed to the (time-variant) master data can be recorded alongside the master data itself. As historic business requirements no longer apply to the data warehouse, it is the current and future metadata that hold the highest value. When time variance is properly implemented then future effective dates for metadata allow the business to set up changes to a data warehouse, which will become effective at a point in time in the future. This improves the process for introducing changes, as a new version of the metadata simply becomes effective when the time of its effective date is reached (rather than requiring the IT function to stop the system, unload/reload the data according to a new version of the database and start the system again).

Data Warehouse Lifecycle Management – a Summary

Data Warehouse Lifecycle Management brings information systems back in line with evolving business requirements, by decreasing data warehouse cycle time. A rapid iteration approach for each data warehouse, deployed piecemeal in a federation, enables the introduction of enterprise-level business performance management at low risk.

Rapid Iteration and Federation require Adaptive Data Warehouses – which are built around a generic storage model, holding time-variant data. Adaptive data warehouses are driven from a real-world business model, putting this and the master data in the hands of the business users rather than IT specialists.

Data Warehouse Lifecycle Management in Action

Kalido's approach to Data Warehouse Lifecycle Management is already powering successful projects for major global enterprises. Sample successes include:

***Shell OP**, the various Oil Products businesses within the Royal Dutch/Shell Group needed to get a standardized global view of performance, while accommodating local diversity and independently-changing local, regional and global business models and data structures. Using DWLM, Shell OP succeeded in building a flexible, cost-effective solution on an un-precedented scale. The DWLM solution brings together management information to support standardization and segmentation, with global and local views of key business entities such as customers and products. This federative approach permits any number of localizations to co-exist with the common corporate data model, giving a consistent top-down view without forcing a structure on individual operating units.*

*Global FMCG giant, **Unilever**, regularly undertakes mergers and acquisitions, so it needed a DWLM solution that would not require its multiple business models to remain static. Using a DWLM approach, Unilever succeeded in bringing together complex, time-variant data from numerous systems, and is using this to deliver relevant and timely management information directly to business users. The company now has commonality across supply-chain, brand, customer and financial data, all cross-referenced by the same master reference data warehouse, ensuring greater consistency and accuracy of information. The solution has made a substantial contribution to savings in procurement, and expanded Unilever's ability to view the historic and projected performance of global brands across financial and non-financial measures. Unilever has also improved its ability to adapt to M&A activity.*

*When Halifax and Bank of Scotland merged to form **HBOS plc**, the board wanted to achieve cost savings by integrating procurement data across the whole organization. Conventional wisdom dictated that a custom-built data* *warehouse would be needed, and that HBOS would need to define an end-point very carefully before starting any work. HBOS could not accept these constraints, because the nature of its ongoing business evolution meant that its organizational structures would be changing regularly. Further-more, HBOS needed an operational data warehouse as quickly as possible, since the board of directors wanted to use the cost savings made within the first few months of the merger as proof of its success. The company's decision to opt for DWLM resulted in a highly successful solution which has contributed to substantial procurement cost savings – a key objective of the merger.*

Kalido has field-proven expertise in iterative data warehouse development, and offers unique software that automates the design and management of adaptive data warehouses. Enterprises using Kalido's pre-built adaptive data warehousing technology can significantly reduce project risk, increase the speed of roll-out, and accelerate return on investment from data warehousing.

For more information on how Data Warehouse Lifecycle Management can deliver cost-effective clarity throughout business change, visit http://www.kalido.com

About Kalido

Kalido provides adaptive enterprise data warehousing software to Global 2000 companies. The KALIDO® application suite (KALIDO) delivers consolidated views of enterprise performance and can immediately adapt them to major changes in the business such as mergers and acquisitions, reorganization, market consolidation, or new regulatory requirements. This improves the speed and accuracy of management and financial reporting without the cost and delay of operational system standardization. Kalido customers who have measured the business benefits of their projects have typically found they have derived annual savings of millions, and in some cases tens of millions, of dollars through improved management of their company performance and reduction of IT costs.

With KALIDO, companies can rapidly create and manage adaptive data warehouses and associated master data throughout their lifecycle, benefiting from the software's strategic flexibility, low cost and speed of deployment. An independently audited study shows that KALIDO saves as least 55 percent in ownership costs compared to custom-built approaches, a figure surpassed by real-life customer experiences. A typical KALIDO data warehouse implementation takes 2-5 months, as opposed to 9-18 months for conventional methods.

Kalido customers include some of the largest companies in the world, such as BP, Cadbury Schweppes, HBOS plc, InBev, Intelsat, Owens Corning, Philips, Royal Dutch/Shell Group of Companies (Shell) and Unilever. These companies and many others use Kalido's award-winning software in over 100 countries for their enterprise-wide data warehousing and master data management projects.

A privately-held company, Kalido is headquartered in Burlington, Mass. and London, UK and has regional sales offices throughout the United States, United Kingdom, and France. More information about Kalido can be found at: http://www.kalido.com.

For more information please contact us:

www.kalido.com
info@kalido.com

www.kalido.com

Kalido
25 Burlington Mall Road
Burlington, MA 01803
United States
Tel: +1 781 305 2300

Kalido
1 The Strand
London
WC2N 5AB
United Kingdom
Tel: +44 (0) 20 7484 2200

Version No: WP-DWLM-ENG-1204

KALIDO